Scientific & Technical Information Center
USPTO

STIC Search Report
EIC 2100

STIC Database Tracking Number: 132166

TO: James Tang
Location: 5C18
Art Unit : 2122
Friday, September 10, 2004

Case Serial Number: 09/710948

From: David Holloway
Location: EIC 2100
PK2-4B30
Phone: 308-7794

david.holloway@uspto.gov

Search Notes

Dear Examiner Tang,

Attached please find your search results for above-referenced case.
Please contact me if you have any questions or would like a re-focused search.

David

sira
Search and Information
Resources Administration

From: Unknown@Unknown.com
Sent: Thursday, September 09, 2004 6:18 PM
To: STIC-EIC2100
Subject: Generic form response

ResponseHeader=Commercial Database Search Request

AccessDB#= 132166

LogNumber= 40

Searcher= ______________

SearcherPhone= ______________

SearcherBranch= ______________

MyDate=Thu Sep 9 18:17:38 EDT 2004

submitto=STIC-EIC2100@uspto.gov

Name=James Tang

Empno=79879

Phone=305-4866

Artunit=2122

Office=5C18

Serialnum=09/710,948

Earliest=11/13/2000

OtherDate=

Format1=paper

Already1=USP

APPENDIX "AA"

What is claimed is:

1. (previously presented) A method of identifying a global breakpoint for debugging computer software, said method including the steps of:

receiving a file name for an executable image file, wherein the executable image file is loaded in memory of a computer system and the global breakpoint is to be placed in the image for executing by the computer system;

receiving a symbol expression for a location in the executable image file where the global breakpoint is to be placed;

passing the symbol expression and the file name to a first operating system module running on the computer system;

receiving a file offset corresponding to the symbol location from the first operating system module;

passing the file name for the executable image file to a second operating system module;

receiving a file identifier from the second operating system module, wherein the file identifier is used by the operating system for uniquely identifying the executable file in the computer system memory; and

representing said global breakpoint in code of said software using the received file identifier of the executable image file and the received offset in said executable file.

2. (original) The method according to claim 1, wherein said file identifier is a file name.

3. (previously presented) The method according to claim 1, wherein said file identifier is an inode of a Unix operating system.

4. (original) The method according to claim 1, wherein said file identifier is a file control block of a non-Unix operating system.

5. (original) The method according to claim 1, further including the step of resolving a virtual address of said code to said file identifier and said offset.

Already2=DWPI

Already3=EPO

Already10=

FFYes=Yes

Searchtopic=Use of operating system modules to determine actual memory locations for instruction of an executable image loaded in a target computer system and inserting these location in the executable image.

Comments=

send=SEND

```
Set     Items   Description
S1     342941   OPERATING()SYSTEM OR OS OR KERNEL OR OS2 OR LINUX OR UNIX
S2      56439   DEBUG? OR DE()(BUGGER OR BUGGING OR BUG OR BUGS) OR BREAKP-
             OINT OR EMULATOR?
S3         67   EXECUTABLE()IMAGE?
S4    5195812   EMBED? OR WITHIN OR INTEGRAT? OR INTEGRAL OR BUILT()"IN"
S5         14   (GLOBAL OR UNIVERSAL)(N)(BREAKPOINT? OR BREAK()POINT? ?)
S6         14   S1 AND S3
S7        759   S2(N)S4
S8         43   S1 AND S7
S9         71   S8 OR S6 OR S5
S10        64   RD (unique items)
S11        56   S10 NOT PY>2000
S12        56   S11 NOT PD>20001113
File   8:Ei Compendex(R) 1970-2004/Aug W5
         (c) 2004 Elsevier Eng. Info. Inc.
File  35:Dissertation Abs Online 1861-2004/Aug
         (c) 2004 ProQuest Info&Learning
File 202:Info. Sci. & Tech. Abs. 1966-2004/Sep 09
         (c) 2004 EBSCO Publishing
File  65:Inside Conferences 1993-2004/Sep W1
         (c) 2004 BLDSC all rts. reserv.
File   2:INSPEC 1969-2004/Aug W5
         (c) 2004 Institution of Electrical Engineers
File  94:JICST-EPlus 1985-2004/Aug W2
         (c)2004 Japan Science and Tech Corp(JST)
File 111:TGG Natl.Newspaper Index(SM) 1979-2004/Sep 10
         (c) 2004 The Gale Group
File 233:Internet & Personal Comp. Abs. 1981-2003/Sep
         (c) 2003 EBSCO Pub.
File   6:NTIS 1964-2004/Aug W4
         (c) 2004 NTIS, Intl Cpyrght All Rights Res
File 144:Pascal 1973-2004/Aug W5
         (c) 2004 INIST/CNRS
File 434:SciSearch(R) Cited Ref Sci 1974-1989/Dec
         (c) 1998 Inst for Sci Info
File  34:SciSearch(R) Cited Ref Sci 1990-2004/Sep W1
         (c) 2004 Inst for Sci Info
File  62:SPIN(R) 1975-2004/Jul W2
         (c) 2004 American Institute of Physics
File  99:Wilson Appl. Sci & Tech Abs 1983-2004/Aug
         (c) 2004 The HW Wilson Co.
File  95:TEME-Technology & Management 1989-2004/Jun W1
         (c) 2004 FIZ TECHNIK
```

12/5/1 (Item 1 from file: 8)
DIALOG(R)File 8:Ei Compendex(R)

04823134 E.I. No: EIP97093836950

Title: Implementation of checkpoints mechanism in PVM parallel Debugging environment

Author: Huang, Ning; Jin, Maozhong; Ma, Wenbin
Corporate Source: Beijing Univ of Aeronautics and Astronautics, Beijing, China

Source: Beijing Hangkong Hangtian Daxue Xuebao/Journal of Beijing University of Aeronautics and Astronautics v 23 n 1 Feb 1997. p 93-97
Publication Year: 1997
CODEN: BHHDE8 ISSN: 1001-5965
Language: Chinese
Document Type: JA; (Journal Article) Treatment: T; (Theoretical)
Journal Announcement: 9711W2

Abstract: A checkpoint mechanism is studied which is used widely in process migrating, load balancing and fault tolerance in parallel programs. An implementation of the checkpoint mechanism in a parallel debugger which uses **global break - points** to ensure the consistence of checkpoints is presented, and the implementation method in PVM environment is discussed, in order to provide a method to debug PVM parallel programs quickly and conveniently. (Edited author abstract) 3 Refs.

Descriptors: *Computer debugging; Fault tolerant computer systems; Parallel processing systems; Computer programming
Identifiers: Checkpoints; Parallel virtual machines
Classification Codes:
723.1 (Computer Programming); 722.4 (Digital Computers & Systems)
723 (Computer Software); 722 (Computer Hardware)
72 (COMPUTERS & DATA PROCESSING)

12/5/3 (Item 3 from file: 8)
DIALOG(R)File 8:Ei Compendex(R)

04431423 E.I. No: EIP96063221389

Title: Event and state-based debugging in TAU: a prototype

Author: Shende, Sameer; Cuny, Janice; Hansen, Lars; Kundu, Joydip; McLaughry, Stephen; Wolf, Odile

Corporate Source: Univ of Oregon, Eugene, OR, USA

Conference Title: Proceedings of the SPDT'96: SIGMETRICS Symposium on Parallel and Distributed Tools

Conference Location: Philadelphia, PA, USA Conference Date: 19960522-19960523

Sponsor: ACM

E.I. Conference No.: 44815

Source: Proceedings of the SPDT: SIGMETRICS Symposium on Parallel and Distributed Tools 1996. ACM, New York, NY, USA. p 21-30

Publication Year: 1996

CODEN: 002380

Language: English

Document Type: CA; (Conference Article) Treatment: G; (General Review); T; (Theoretical)

Journal Announcement: 9608W3

Abstract: Parallel programs are complex and often require a multilevel debugging strategy that combines both event- and state-based debugging. We report here on preliminary work that combines these approaches within the TAU program analysis environment for pC plus plus . This work extends the use of event-based modeling to object-parallel languages, provides an alternative mechanism for establishing meaningful **global breakpoints** in object-oriented languages, introduces the TAU program interaction and control infrastructure, and provides an environment for the assessment of mixed event- and state-based strategies. (Author abstract) 23 Refs.

Descriptors: *Program debugging; Software prototyping; Computer simulation; Object oriented programming; Codes (symbols); Coding errors; Visualization

Identifiers: Parallel program; Ariadne debugger

Classification Codes:

723.1 (Computer Programming); 723.5 (Computer Applications); 723.2 (Data Processing)

723 (Computer Software)

72 (COMPUTERS & DATA PROCESSING)

12/5/35 **(Item 8 from file: 233)**
DIALOG(R)File 233:Internet & Personal Comp. Abs.

00210435 90CP02-003

Advanced debugging techniques Employ these sophisticated strategies to reduce debugging time

Intersimone, David

Computer Language , February 1, 1990 , v7 n2 p59-67, 7 Pages
ISSN: 0749-2839
Languages: English
Document Type: Feature Articles and News

C program

Geographic Location: United States

Presents several advanced debugging techniques which locate bugs quickly. Discusses the use of code patch to test fix, **global breakpoints** , conditional **global breakpoints** , instrumenting code, and complex break conditions. Includes six program listings. (jb)

Descriptors: Debugging; Strategy; Bugs; Programming Instruction; Program Listing

12/5/42 (Item 2 from file: 99)
DIALOG(R)File 99:Wilson Appl. Sci & Tech Abs

1219094 H.W. WILSON RECORD NUMBER: BAST95014553
Debugging embedded **software**
Cipriani, Dave;
Electronic Design v. 43 (Jan. 23 '95) p. 103-4+
DOCUMENT TYPE: Feature Article ISSN: 0013-4872 LANGUAGE: English
RECORD STATUS: New record

ABSTRACT: Part of a special section on engineering software for software design engineers. A cost-effective environment for debugging can be created by matching the correct tools with the job. The cornerstone of the software-development environment is the source-level debugger. A developer's window into the target system, the debugger offers an abstracted view of program execution. The debugger's minimum requirements include the ability to load code, monitor variables, alter memory using symbolic names and high-level expressions, display source code, and set breakpoints. **Debuggers integrated** with an **operating system** have knowledge of the **operating - system** data structures, enabling the engineer to monitor when tasks are entered and exited. Choosing a debugger that can control different execution vehicles provides a consistent interface throughout product development. The advantages and disadvantages of debugging tools such as simulators, full-featured emulators, N-wire emulators, logic analyzers, and ROM monitors are considered.

DESCRIPTORS: Emulators (Computers); Software development--Costs; Debug tools (Computer programs);

12/5/55 **(Item 11 from file: 95)**
DIALOG(R)File 95:TEME-Technology & Management

00586240 E92073284010
Specification and detection of global breakpoints **in distributed systems**
(Spezifikation und Feststellung globaler Breakpoints in verteilten Systemen
)
Haban, D; Zhou, S; Maurer, D; Wilhelm, R
Univ. des Saarlandes, Saarbruecken, D
1991
Document type: Report Language: English
Record type: Abstract

ABSTRACT:
Writing and debugging asynchronous parallel programs is a painful experience. Verification methods are even less developed and of less practical use than in the sequential case. Static analyzers and dynamic debuggers for, such programs are rare. Setting breakpoints are important features of any debugging system. The authors will refer to these breakpoints as **global breakpoints** . Due to the inherent features of distributed systems, these systems are characterized by lack of adequate central control, precise global time and accurate global state. Thus, specification of **global breakpoints** , their detection and finally halting the distributed system in a meaningful global state are major problems. They propose a specification language for **global breakpoints** and describe its logical semantics. The formal model of a detection mechanisms and its implementation are given and solutions to the halting of distributed systems are discussed. The distributed debugging system has been implemented and successfully used to debug distributed programs running on six M68000 computers.

DESCRIPTORS: PARALLEL PROGRAMMING; PROGRAM DEVELOPMENT; PROGRAM VERIFICATION; COMPUTER ARCHITECTURE; PROCESSORS; PARALLEL PROCESSORS; TEST AID PROGRAM
IDENTIFIERS: Parallelprogrammierung; Debugger; Breakpoint

```
Set     Items     Description
S1      37034     OPERATING()SYSTEM OR OS OR KERNEL OR OS2 OR LINUX OR UNIX
S2      11924     DEBUG? OR DE()(BUGGER OR BUGGING OR BUG OR BUGS) OR BREAKP-
               OINT OR EMULATOR?
S3         36     EXECUTABLE()IMAGE?
S4    1791609     EMBED? OR WITHIN OR INTEGRAT? OR INTEGRAL OR BUILT()"IN"
S5          2     (GLOBAL OR UNIVERSAL)(N)(BREAKPOINT? OR BREAK()POINT? ?)
S6       1205     1 AND S2 AND S4
S7          4     S1 AND (S3 OR S5)
S8         19     S2(2N)S4 AND S1
S9         23     S7 OR S8
S10        24     S9 OR S5
S11        24     IDPAT (sorted in duplicate/non-duplicate order)
S12        23     IDPAT (primary/non-duplicate records only)
File 347:JAPIO Nov 1976-2004/May(Updated 040903)
         (c) 2004 JPO & JAPIO
File 350:Derwent WPIX 1963-2004/UD,UM &UP=200457
         (c) 2004  Thomson Derwent
```

12/5/7 (Item 7 from file: 350)

DIALOG(R)File 350:Derwent WPIX

014736446 **Image available**

WPI Acc No: 2002-557150/200259

XRPX Acc No: N02-441058

Global breakpoint **insertion method for debugging software, involves detecting private copy of page after reading from memory and inserting** global breakpoint **in detected private copy**

Patent Assignee: INT BUSINESS MACHINES CORP (IBMC)

Inventor: BHATTACHARYA S; KRISHNAMOORTHY A; SANGAVARAPU V K

Number of Countries: 001 Number of Patents: 001

Patent Family:

Patent No	Kind	Date	Applicat No	Kind	Date	Week	
US 20020073402	A1	20020613	US 2000732342	A	20001207	200259	B

Priority Applications (No Type Date): US 2000732342 A 20001207

Patent Details:

Patent No	Kind	Lan	Pg	Main IPC	Filing Notes
US 20020073402	A1		13	G06F-009/44	

Abstract (Basic): US 20020073402 A1

NOVELTY - A **global breakpoint** is inserted in a page containing software code after reading the page in a memory. The **global breakpoint** is inserted in the detected private copy of the page.

DETAILED DESCRIPTION - INDEPENDENT CLAIMS are included for the following:

(1) Computer-implemented apparatus for inserting **global breakpoint** ;

(2) Computer program product for inserting **global breakpoints** ;

(3) **Global breakpoint** removal method;

(4) Computer-implemented apparatus for removing **global breakpoint** ; and

(5) Computer program product of removing **global breakpoint** .

USE - For debugging computer software in **operating system** such as **Linux** e.g. SUN, HP and AIX for computer system, connected to LAN, WAN, Internet, intranet.

ADVANTAGE - Enables handling elegantly, consistently and seamlessly of the problem of inserting and removing **global breakpoints** with minimum overhead, hence facilitates the **OS** to maintain sufficient information effectively.

DESCRIPTION OF DRAWING(S) - The figure shows the block diagram of the **operating system** .

pp; 13 DwgNo 1/3

Date
N6

Title Terms: GLOBE; INSERT; METHOD; DEBUG; SOFTWARE; DETECT; PRIVATE; COPY; PAGE; AFTER; READ; MEMORY; INSERT; GLOBE; DETECT; PRIVATE; COPY

Derwent Class: T01

International Patent Class (Main): G06F-009/44

File Segment: EPI

```
 12/5/17     (Item 17 from file: 350)
DIALOG(R)File 350:Derwent WPIX
(c) 2004  Thomson Derwent. All rts. reserv.

010064945     **Image available**
WPI Acc No: 1994-332656/199441
XRPX Acc No: N94-261196
  Computer system having  integrated  source level  debugging  functions -
  has system management handler executing resume instruction to continue
  execution with interrupted program upon exiting  integrated   debugger
Patent Assignee: INTEL CORP (ITLC  )
Inventor: YUEN D
Number of Countries: 001  Number of Patents: 001
Patent Family:
Patent No     Kind   Date     Applicat No     Kind   Date     Week
US 5357628     A   19941018  US 92858301     A   19920325  199441  B
                             US 93138894     A   19931019

Priority Applications (No Type Date): US 92858301 A 19920325; US 93138894 A
  19931019
Patent Details:
Patent No  Kind Lan Pg   Main IPC    Filing Notes
US 5357628    A      7 G06F-011/00   Cont of application US 92858301

Abstract (Basic): US 5357628 A
        Debugging is performed under SMM with the integrated   debugger
    which is stored with the SMI handler in the SMRAM and given control
    after the SMI handler has gotten control and determined in its initial
    processing that the SMI handler has gotten control as a result of a SMI
    triggered by a debugging request.
        The SMI handler gets control after the computer system is put into
    SMM in response to the SMI. Upon exiting the integrated   debugger ,
    the SMI handler executes the RSM instruction to continue execution with
    the interrupted program.
        ADVANTAGE - Debugging may be performed with the actual hardware in
    its normal operating speed, and yet debugging functions and usability
    matching or exceeding that of a software emulator may be provided.
    Additionally, debugging may be performed in a manner that is
    transparent to the  operating   system  and the application programs.
        Dwg.2/2
Title Terms: COMPUTER; SYSTEM; INTEGRATE; SOURCE; LEVEL; DEBUG; FUNCTION;
  SYSTEM; MANAGEMENT; HANDLE; EXECUTE; RESUME; INSTRUCTION; CONTINUE;
  EXECUTE; INTERRUPT; PROGRAM; EXIT; INTEGRATE
Derwent Class: T01
International Patent Class (Main): G06F-011/00
File Segment: EPI
```

12/5/21 (Item 21 from file: 347)
DIALOG(R)File 347:JAPIO

06513647 **Image available**
METHOD FOR DEBUGGING DEVICE DRIVER

PUB. NO.:	2000-099364 [JP 2000099364 A]
PUBLISHED:	April 07, 2000 (20000407)
INVENTOR(s):	NOGUCHI SEIJI
APPLICANT(s):	MATSUSHITA ELECTRIC IND CO LTD
APPL. NO.:	10-283267 [JP 98283267]
FILED:	September 18, 1998 (19980918)
INTL CLASS:	G06F-011/28

ABSTRACT

PROBLEM TO BE SOLVED: To enable debugging without preparing any dedicated debugging object device driver and changing any internal code by monitoring a request packet to be dispatched from an **operating system** (**OS**) to the debugging object device driver and controlling the output of input/output data.

SOLUTION: When a computer is activated, an **OS** 4 and a debugging object device driver 6 are integrated onto an internal RAM 2. Next, a device driver 5 for **debugging** is **integrated** onto the RAM 2 and during that integrating processing, processing is performed for hooking the entry routine of the debugging object device driver 6. Then, this method uses a device driver 5 for debugging provided with a method for controlling the output of output data by monitoring the request packet to be dispatched from the **OS** 4 to the debugging object device driver 6.

```
 12/5/3     (Item 3 from file: 350)
DIALOG(R)File 350:Derwent WPIX
(c) 2004  Thomson Derwent. All rts. reserv.

015572611     **Image available**
WPI Acc No: 2003-634768/200360
XRPX Acc No: N03-504820
  Java application debugging method in corporate network, internet,
  involves debugging Java code and native language dynamic load libraries
  simultaneously using different application programming interfaces
Patent Assignee: INT BUSINESS MACHINES CORP (IBMC  )
Inventor: EVANS D H; GAY C J; SCHERK A P
Number of Countries: 001  Number of Patents: 001
Patent Family:
Patent No      Kind   Date     Applicat No    Kind   Date      Week
US 20020129337  A1  20020912  US 2001801589    A   20010308  200360  B

Priority Applications (No Type Date): US 2001801589 A 20010308
Patent Details:
Patent No  Kind Lan Pg   Main IPC    Filing Notes
US 20020129337 A1    25 G06F-009/44

Abstract (Basic): US 20020129337 A1
        NOVELTY - The Java code and native language dynamic load libraries
    of application (17) are simultaneously debugged using Java platform
    debugger architecture virtual machine debug application programming
    interface (API) and operating  system debug API, respectively, while
    executing application under Java virtual machine (16).
        DETAILED DESCRIPTION - INDEPENDENT CLAIMS are also included for the
    following:
        (1) computer readable medium storing application debugging program;
    and
        (2) computer.
        USE - For debugging Java application comprising Java code and
    native language dynamic load libraries e.g. C or C++ code, using
    computer software development tools for development of Java application
    program for use in corporate network and internet.
        ADVANTAGE - Provides new functionality such as patching of Java
    variables, reading and writing strings from and to application under
    test and stability for JAVA application program development, testing
    and debugging.
        DESCRIPTION OF DRAWING(S) - The figure shows the block diagram of
    the debugger system within single computer system.
        graphical user interface (11)
        engine (12)
        Java virtual machine (16)
        application(41) interactive code analysis tool probe (17)
        pp; 25 DwgNo 1/15
Title Terms: APPLY; DEBUG; METHOD; NETWORK; DEBUG; CODE; NATIVE; LANGUAGE;
  DYNAMIC; LOAD; SIMULTANEOUS; APPLY; PROGRAM; INTERFACE
Derwent Class: T01
International Patent Class (Main): G06F-009/44
File Segment: EPI
```

```
Set     Items   Description
S1      37034   OPERATING()SYSTEM OR OS OR KERNEL OR OS2 OR LINUX OR UNIX
S2      11924   DEBUG? OR DE()(BUGGER OR BUGGING OR BUG OR BUGS) OR BREAKP-
             OINT OR EMULATOR?
S3         36   EXECUTABLE()IMAGE?
S4    1791609   EMBED? OR WITHIN OR INTEGRAT? OR INTEGRAL OR BUILT()"IN"
S5          2   (GLOBAL OR UNIVERSAL)(N)(BREAKPOINT? OR BREAK()POINT? ?)
S6       1205   1 AND S2 AND S4
S7          4   S1 AND (S3 OR S5)
S8         19   S2(2N)S4 AND S1
S9         23   S7 OR S8
S10        24   S9 OR S5
S11        24   IDPAT (sorted in duplicate/non-duplicate order)
S12        23   IDPAT (primary/non-duplicate records only)
S13        14   AU=(KRISHNA S, OR KRISHNA S?)
S14         5   AU=(SONI M? OR SONI, M?)
S15        90   AU=BHATTACHARYA S?
S16         1   S15 AND S1
S17     86855   MC=(T01-F05G OR T01-J20C OR T01-S03)
S18       224   S17 AND S6
S19        23   S1 AND S2 AND S4 AND S17
S20        15   S19 NOT S11
File 347:JAPIO Nov 1976-2004/May(Updated 040903)
         (c) 2004 JPO & JAPIO
File 350:Derwent WPIX 1963-2004/UD,UM &UP=200457
         (c) 2004  Thomson Derwent
```

Redbooks ▾

Home | Products & services | Support & downloads | My account

Select a country

Redbooks Home

Publications:

- Drafts
- Redbooks
- Redpapers
- Technotes
- Additional materials

Redbook Domains:

- -- Select one -- ▾

Residencies

Workshops

Redbooks on CD

How to buy

About Redbooks

Contact us

Mailing list

Related links:

IBM Publications

Technical Training

Developers

IBM Business Partners

Redbooks Search

Use AND, OR, NOT to separate keywords

New search	executable image
Sort by	relevance ▾ □ Fuzzy search
search tips	Search

0 results found in Redbooks, Redpapers, Drafts and Technotes
To learn more about Redbooks and Redpapers, click here.

0 results found in Residencies

0 results found in Workshops

About IBM | **Privacy** | **Terms of use** | **Contact**

Redbooks Home

Redbooks ▾

Home | Products & services | Support & downloads | My account

Select a country

Redbooks Home

Publications:

- Drafts
- Redbooks
- Redpapers
- Technotes
- Additional materials

Redbook Domains:

- -- Select one -- ▾

Residencies

Workshops

Redbooks on CD

How to buy

About Redbooks

Contact us

Mailing list

Related links:

IBM Publications

Technical Training

Developers

IBM Business Partners

Redpapers Index

Redpapers are technical documents that have been written to address a specific topic. They are not planned to become published redbooks and are not necessarily the result of an ITSO residency.

Note that these documents are not orderable in hardcopy. Since these papers are not published through our normal publishing process, the layouts may vary.

Redpapers - Search Results

executable image Search

Use AND, OR, NOT to separate keywords

0 results found

Back to Index

About IBM | Privacy | Terms of use | Contact

Redbooks Home

```
Set     Items   Description
S1          3   GLOBAL()BREAKPOINT AND EXECUT?()IMAGE?
File 654:US Pat.Full. 1976-2004/Sep 09
         (c) Format only 2004 The Dialog Corp.
```

1/TI/1
DIALOG(R)File 654:(c) Format only 2004 The Dialog Corp. All rts. reserv.

Communications system using rings architecture

1/TI/2
DIALOG(R)File 654:(c) Format only 2004 The Dialog Corp. All rts. reserv.

Method for inserting global breakpoints

1/TI/3
DIALOG(R)File 654:(c) Format only 2004 The Dialog Corp. All rts. reserv.

Superscalar microprocessor employing away prediction structure

```
*** DIALINDEX search results display in an abbreviated ***
*** format unless you enter the SET DETAIL ON command. ***
?set files all
   You have 558 files in your file list.
   (To see banners, use SHOW FILES command)
?s global()breakpoint and execut?()image?

Your SELECT statement is:
   s global()breakpoint and execut?()image?

           Items   File
           -----   ----
       Examined  50 files
       Examined 100 files
       Examined 150 files
       Examined 200 files
       Examined 250 files
       Examined 300 files
       Examined 350 files
       Examined 400 files
               3   654: US Pat.Full._1976-2004/Sep 09
       Examined 450 files
       Examined 500 files
       Examined 550 files

   1 file has one or more items; file list includes 558 files.
   One or more terms were invalid in 2 files.
```